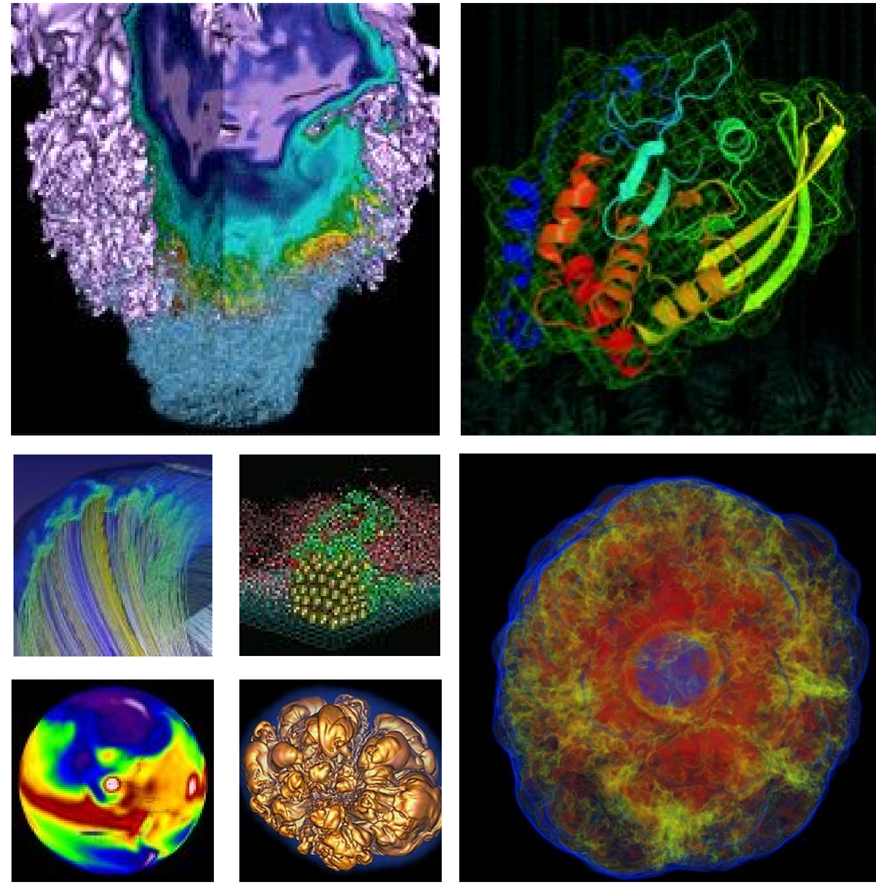
Running jobs at NERSC (Cori, Edison)



Steve Leak, NERSC
User Engagement Group

Key Points

- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help!** **consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

Today's Agenda

- **How jobs work**
- **What *must* I request?**
- **My first job**
- **What else *can* I request?**
- **Where is my job?**
- **Working interactively**
- **Getting through the queue faster**
- **How usage is charged**
- **Which cores are running what? (advanced)**
- **Workflows - job arrays and dependencies**

Key Points

- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - ubmit a job
  - ptionally MPI) application within a job
  - queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help!** [**consult@nersc.gov**](mailto:consult@nersc.gov)

All of this is on the web!

[**www.nersc.gov/users/**computational-systems/{cori,edison}/**running-jobs/**](www.nersc.gov/users/computational-systems/{cori,edison}/running-jobs/)

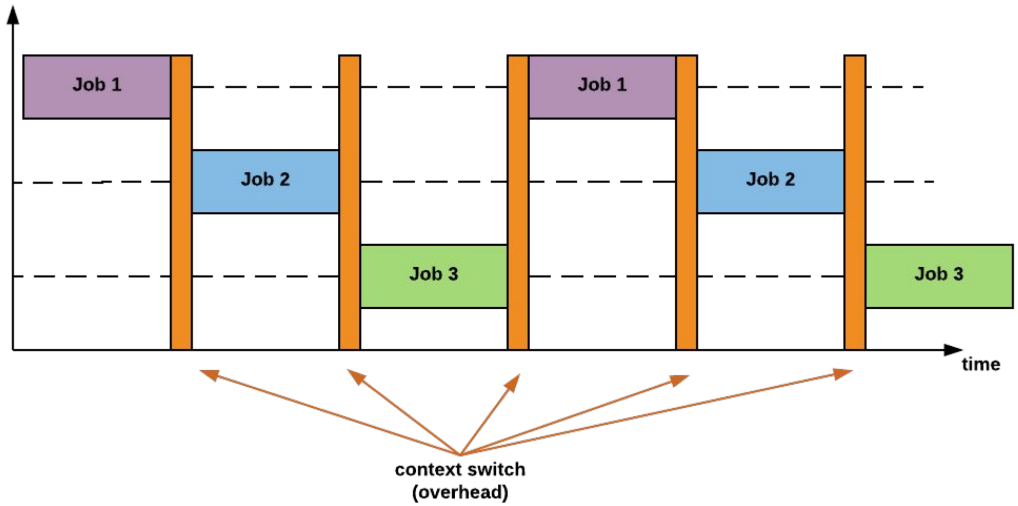
Today's Agenda

- **How jobs work**
- What *must* I request?
- My first job
- What else *can* I request?
- Where is my job?
- Working interactively
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

How jobs work

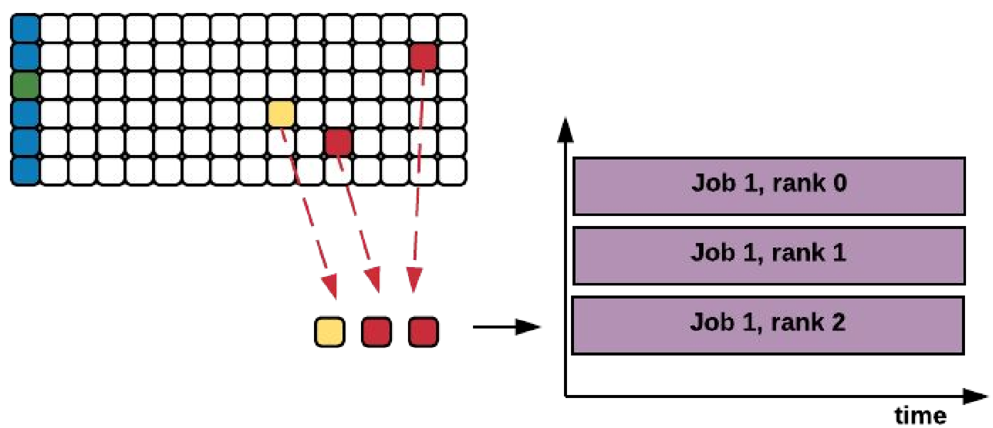
Desktop / login node

- Timeslicing
  - core shared by multiple tasks
  - Works when the computer is mostly waiting for you



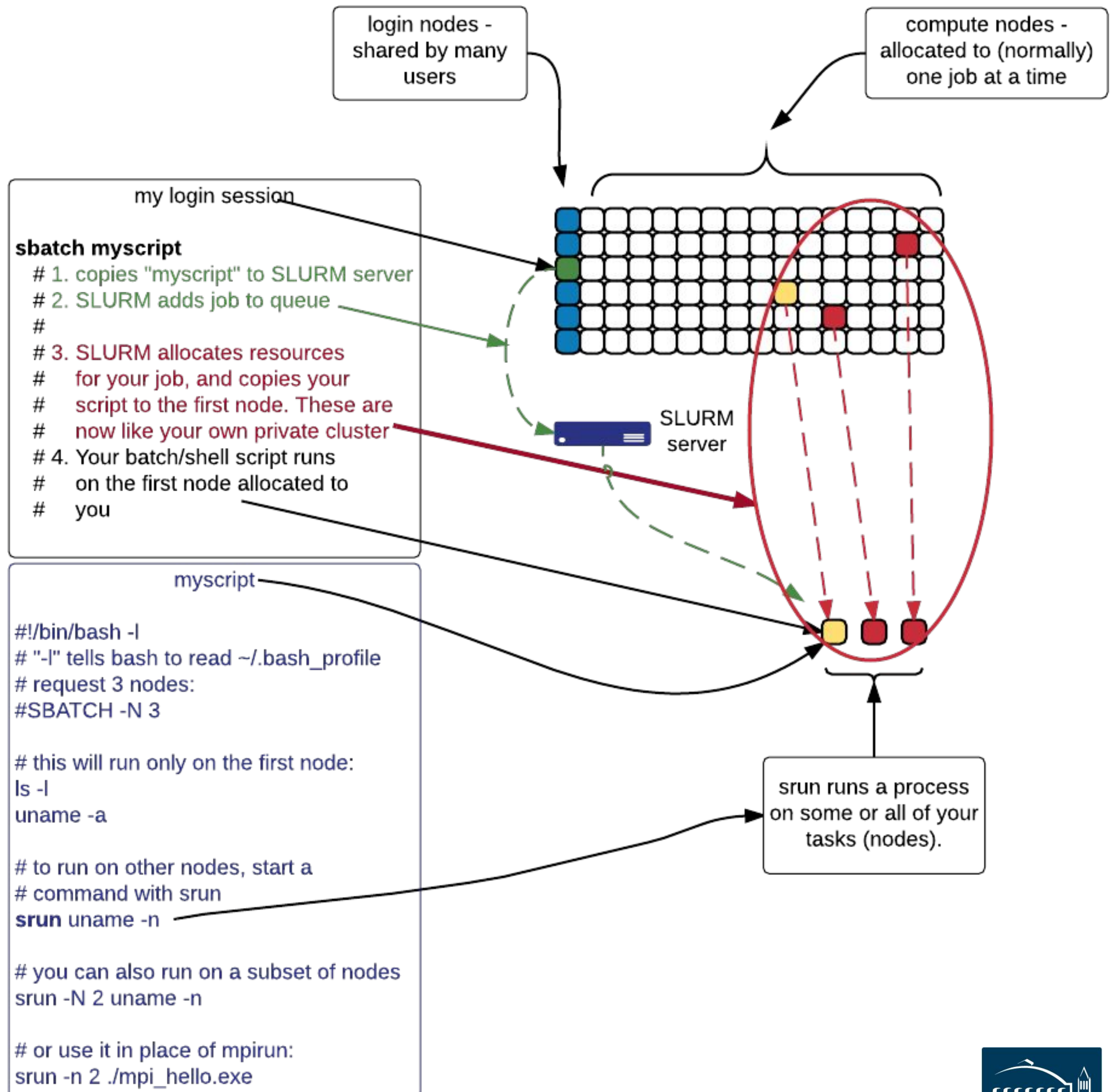
HPC

- You are waiting for the computer
- Subset of pooled resources dedicated to one job



How jobs work

- **Start on login node**
  - shared by many users, not for computational work
- **Access compute nodes with sbatch or salloc**
- **Batch script**
  - Copied to queue
  - Has directives for SLURM, and shell commands to perform on first compute node
- **Access your other allocated nodes with srun**
- **stdout, stderr saved to file**
  - (when running in batch mode)



Key Points

- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help! consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

Today's Agenda

- How jobs work
- **What *must* I request?**
- My first job
- What else *can* I request?
- Where is my job?
- Working interactively
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

What *must* I request?

What the batch system needs to know:

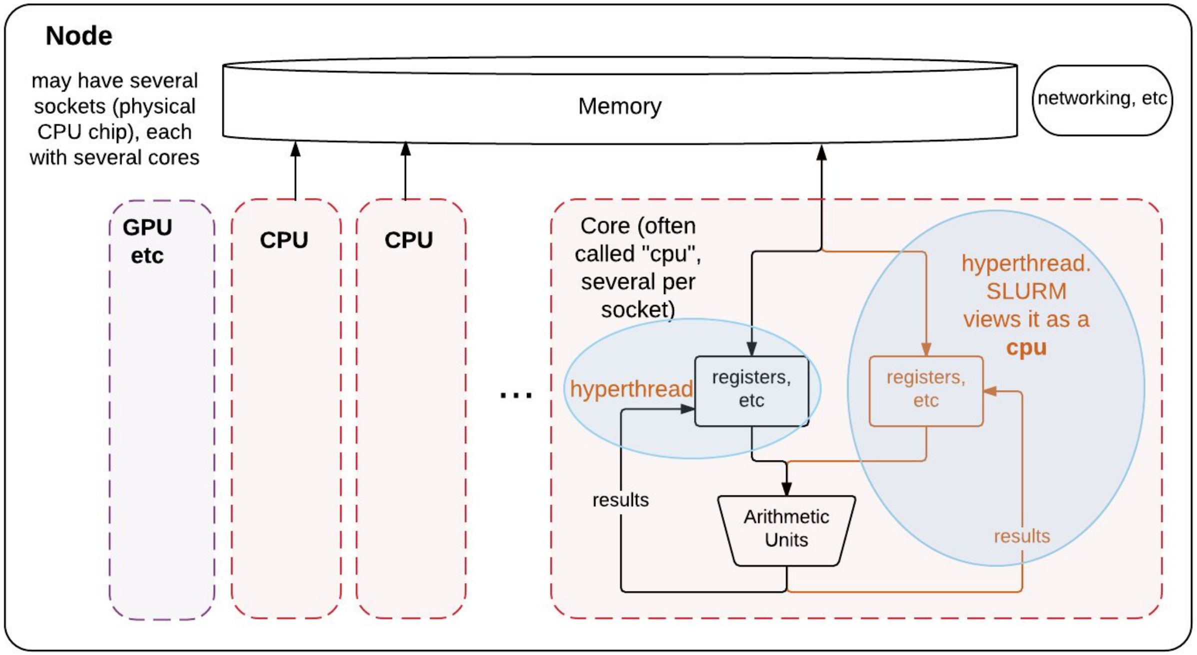
- **How many nodes (or CPUs) does this job need?**
  - Mostly NERSC allocates and charges by the node
  - Jobs needing no more than half of one node - and willing to coexist - may request CPUs in the “shared” partition
- **For how long does it need them?**
  - Wallclock time limit

NERSC-specific extras:

- **What type of CPU?**
  - KNL or Xeon (haswell/ivybridge)?
- **Which filesystems will this job use?**

Nodes, cores, CPUs, threads, tasks - some definitions

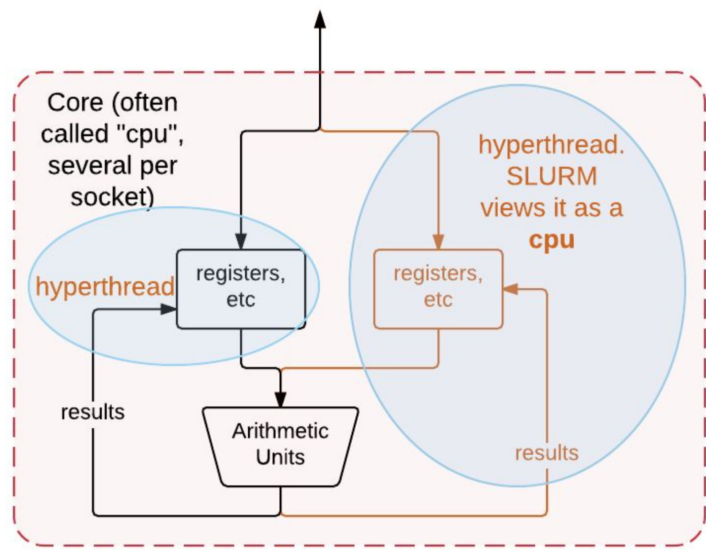
- ***Node* is the basic unit of allocation at NERSC**
  - Think “one host” or “one server”
  - Single memory space, multiple CPU cores (24 or 32 or 68 ...
    - And a core might support hyperthreading

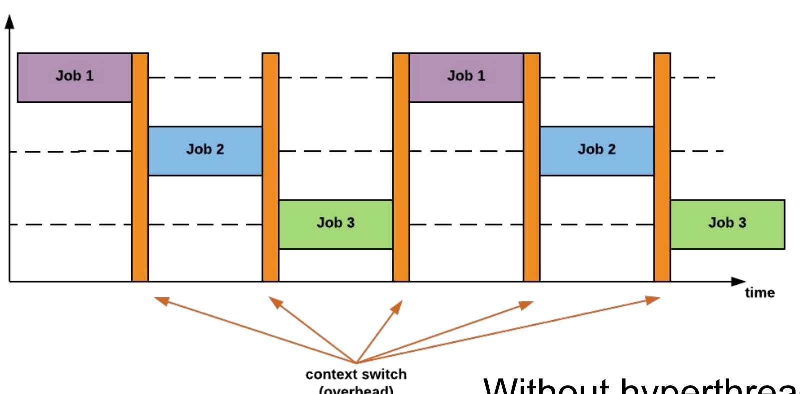


Nodes, cores, CPUs, threads, tasks - some definitions

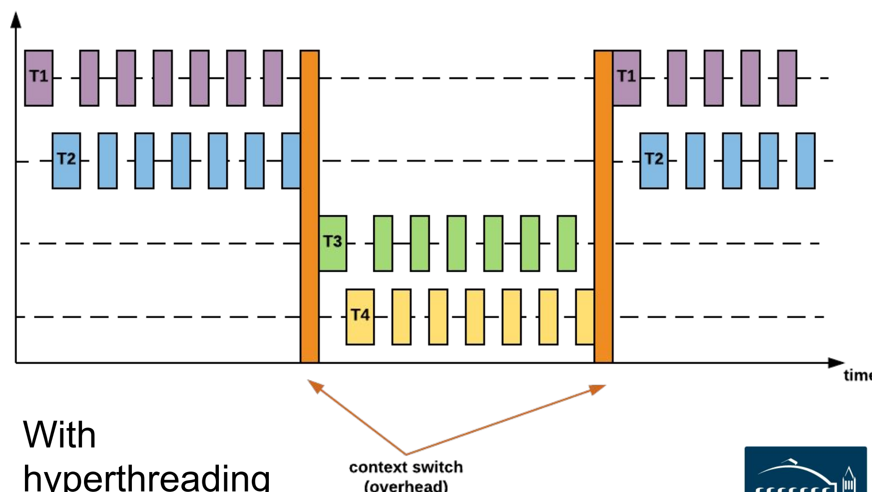
Hyperthreading

- Fast timeslicing
  - Good when arithmetic units frequently wait on memory
- Core holds state of 2 (4 on KNL) processes, they share arithmetic units
- **SLURM views each hyperthread as a *CPU***
- But most HPC jobs perform best when not sharing a core!
- Usually best to reserve 2 CPUs / core





Without hyperthreading

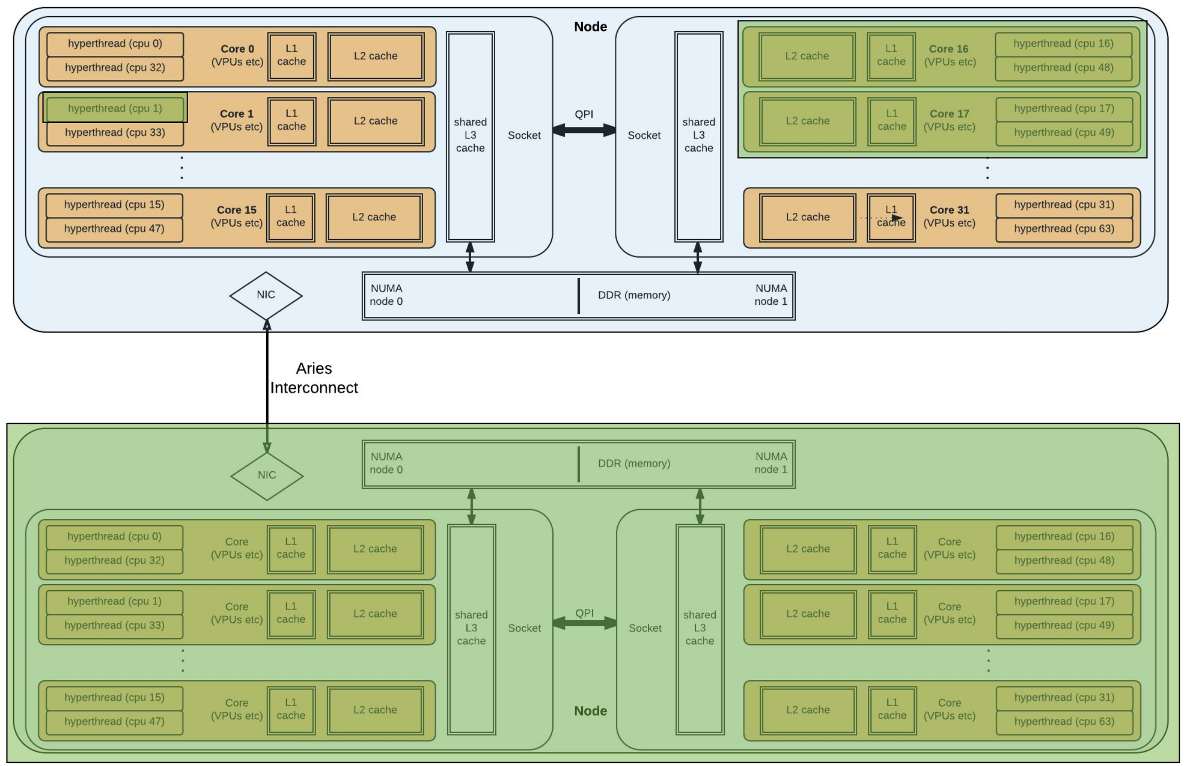


With hyperthreading

Nodes, cores, CPUs, threads, tasks - some definitions

- **A SLURM *task* is a reservation of CPUs and memory, up to one full node**
  - A *job* has many tasks

`srun -n <ntasks> ..`



- **Eg: 3 possible tasks on 2 nodes**

What *must* I request?

What the batch system needs to know:

- **How many nodes (or CPUs) does this job need?**
  - Mostly NERSC allocates and charges by the node
  - Jobs needing no more than half of one node - and willing to coexist - may request CPUs in the “shared” partition
- **For how long does it need them?**
  - Wallclock time limit

NERSC-specific extras:

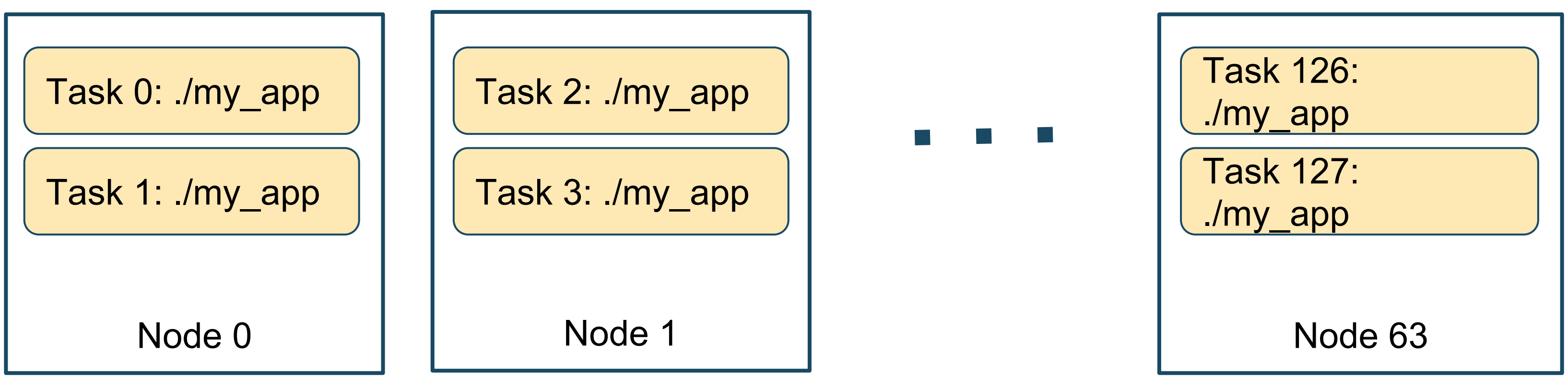
- **What type of CPU?**
  - KNL or Xeon (haswell/ivybridge)?
- **Which filesystems will this job use?**

How many nodes?

- **Optimal number of MPI ranks depends on your application and problem size (trial and error)**
  - NERSC nodes mostly have 2-4 GB memory / core,
    64-128 GB memory / node, and 24, 32 or 68 cores / node
  - You will need some minimum number of nodes for enough memory to hold the problem size
- **Some applications are also multithreaded (OpenMP)**
  - Trial and error to discover optimal number of OpenMP threads per MPI rank and of MPI ranks per node
- ***Most* HPC applications run best with a full core (not just a hyperthread) dedicated to each OpenMP thread**

Requesting nodes or tasks

```
#SBATCH -N 64           # request 64 nodes
srun -N 32 ./my_app     # start ./my_app on 32 of them
                        # (default: 1 per node)
srun -n 128 ./my_app    # start 128 instances of ./my_app,
                        # across my 64 nodes (default is
                        # to evenly distribute them in
                        # block fashion)
```



One MPI rank generally corresponds to one SLURM Task

Requesting tasks and cores

```
#SBATCH -n 1024      # request sufficient nodes for 1024
#SBATCH -c 2         # SLURM tasks (MPI ranks), with 2 cpus
                     # (hyperthreads), ie 1 full Xeon core,
                     # per task
```

```
#SBATCH -n 1024                # request nodes for 1024
#SBATCH --ntasks-per-node=24   # MPI ranks, with no more
                               # than 24 ranks on any node
```

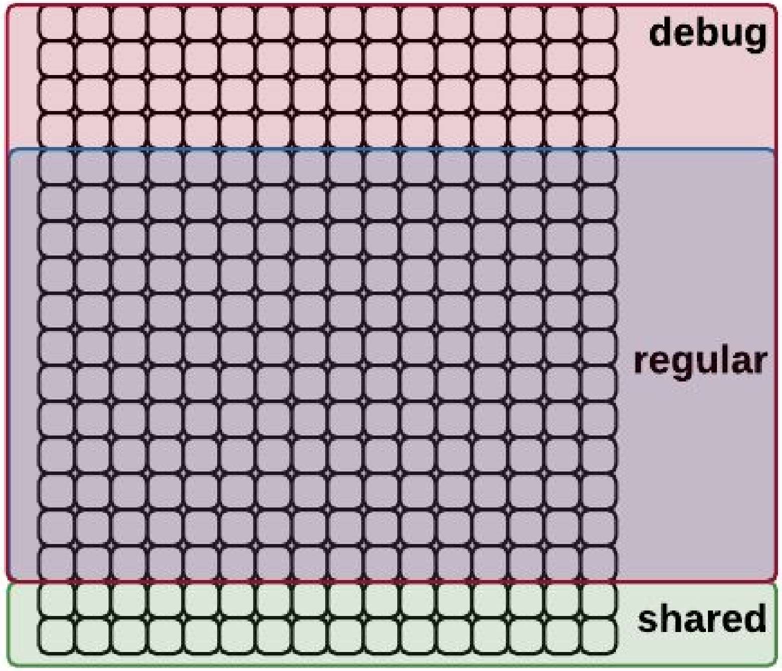
```
#SBATCH -n 1024      # request nodes for 1024 MPI ranks,
#SBATCH -c 8         # with 8 cpus (ie 4 Xeon cores or
                     # 2 KNL cores) per rank
                     # (suitable with OMP_NUM_THREADS=4)
```

Partitions

- **A *partition* is the subset of nodes your job can use**
  - Each partition has rules about size of job that can use it
- **NERSC partitions**
  - **debug** - for small, short jobs needing quick turnaround (up to 64 or 512 nodes, up to 30 minutes - default partition for jobs that fit)
  - **regular** - for most real work (up to whole partition, or up to 4 days)
  - **shared** - for jobs needing half-a-node or less (up to 2 days)

Request partition with (eg):

```
#SBATCH -p debug
```



What *must* I request?

What the batch system needs to know:

- How many nodes (or CPUs) does this job need?
  - Mostly NERSC allocates and charges by the node
  - Jobs needing no more than half of one node - and willing to coexist - may request CPUs in the “shared” partition
- **For how long does it need them?**
  - Wallclock time limit

NERSC-specific extras:

- What type of CPU?
  - KNL or Xeon (haswell/ivybridge)?
- Which filesystems will this job use?

Requesting time

```
#SBATCH -t 30          # 30 minutes
#SBATCH -t 30:00       # 30 minutes
#SBATCH -t 1:00:00     # 1 hour
#SBATCH -t 1-0         # 1 day
#SBATCH -t 1-12        # 1.5 days
```

- **Wallclock time, ie real elapsed time**
- **After this much time, SLURM can kill this job**

What *must* I request?

What the batch system needs to know:

- **How many nodes (or CPUs) does this job need?**
  - Mostly NERSC allocates and charges by the node
  - Jobs needing no more than half of one node - and willing to coexist - may request CPUs in the “shared” partition
- **For how long does it need them?**
  - Wallclock time limit

NERSC-specific extras:

- **What type of CPU?**
  - KNL or Xeon (haswell/ivybridge)?
- **Which filesystems will this job use?**

What type of CPU?

- **Cori has 2 types of nodes (haswell, knl)**
- **The KNL nodes have multiple modes for:**
  - MCDRAM (flat, cache, hybrid)
  - NUMA arrangement of a node (quad, hemi, a2a, snc4, snc2)
- **No default (you *must* specify)**
- **Specify via SLURM constraint option, -C**

```
#SBATCH -C haswell
#SBATCH -C knl,quad,cache
#SBATCH -C knl,quad,flat
```

- **Supported also on Edison (ivybridge)**

Which filesystems?

- **Context: NERSC provides several filesystems**
  - Most jobs use 1 or 2 of them ($SCRATCH or $PROJECT)
- **Problem: sometimes a filesystem is unavailable**
  - Maintenance, or a failure
  - Jobs that try to use an unavailable filesystem tend to crash
    - Maybe after several days in queue!
- **Solutions:**
  - Let them crash
  - Stop *all* jobs when *any* filesystem is unavailable
  - **Require that jobs specify which filesystems they need, and stop only those jobs (from starting)**

Which filesystems?

- **Specify filesystems via SLURM "license" feature:**
```
#SBATCH -L SCRATCH
#SBATCH -L scratch1,project
```
- **Not used for $HOME**
  - (*Everything* needs $HOME)
- **Filesystems you can specify:**
  - SCRATCH (scratch1, scratch2, cscratch1)
  - project
  - projecta, projectb, dna
  - scratch3 (Edison only)
  - cscratch1 (either cluster)

Today's Agenda

- How jobs work
- What *must* I request?
- **My first job**
- What else *can* I request?
- Where is my job?
- Working interactively
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

My first job

A SLURM job script has two sections:

1. **Directives telling SLURM what you would like it to do with this job**
2. **The script itself - shell commands to run on the first compute node**

```
elvis@nersc:~> vi myscript.q
```

```
#!/bin/bash -l

#SBATCH -t 00:30:00
#SBATCH -N 2
#SBATCH --license=SCRATCH

export RUNDIR=$SCRATCH/run-$SLURM_JOBID
mkdir -p $RUNDIR
cd $RUNDIR

srun -n 4 bash -c 'echo "Hello, world, from node $(hostname)"'
```

```
elvis@nersc:~> sbatch -C $CRAY_CPU_TARGET myscript.q
Submitted batch job 2774102
```

My first job

A SLURM job script has two sections:

1. **Directives telling SLURM what you would like it to do with this job**
2. **The script itself - shell commands to run on the first compute node**

How many nodes?

For how long?

No --qos == normal priority and cost

```
elvis@nersc:~> vi myscript.q
```

```
#!/bin/bash -l

#SBATCH -t 00:30:00
#SBATCH -N 2
#SBATCH --license=SCRATCH

export RUNDIR=$SCRATCH/run-$SLURM_JOBID
mkdir -p $RUNDIR
cd $RUNDIR

srun -n 4 bash -c 'echo "Hello, world, from node $(hostname)"'
```

```
elvis@nersc:~> sbatch -C $CRAY_CPU_TARGET myscript.q
Submitted batch job 2774102
```

$SCRATCH filesystem

Xeon nodes on current cluster (set by craype-{haswell,ivybridge} module)

Note: cannot use env vars in directives - but directives have equivalent command-line option

My first job

A SLURM job script has two sections:

1. **Directives telling SLURM what you would like it to do with this job**
2. **The script itself - shell commands to run on the first compute node**

```
elvis@nersc:~> vi myscript.q
```

Make starting environment like my login environment

Run from $SCRATCH

Start 4 tasks across my nodes

```
#!/bin/bash -l

#SBATCH -t 00:30:00
#SBATCH -N 2
#SBATCH --license=SCRATCH

export RUNDIR=$SCRATCH/run-$SLURM_JOBID
mkdir -p $RUNDIR
cd $RUNDIR

srun -n 4 bash -c 'echo "Hello, world, from node $(hostname)"'
```

```
elvis@nersc:~> sbatch -C $CRAY_CPU_TARGET myscript.q
Submitted batch job 2774102
```

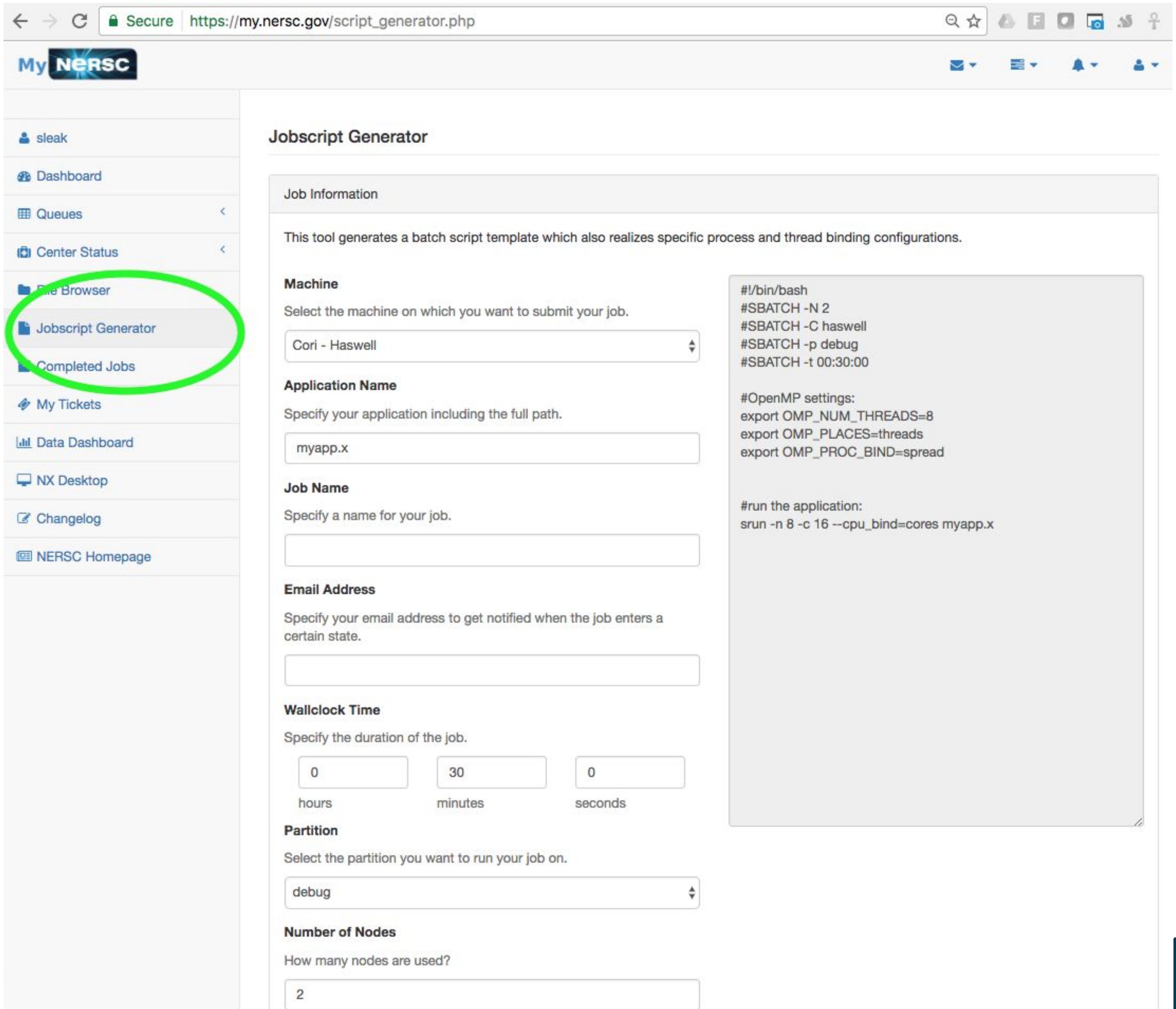
"sbatch" submits a job script

Key Points

- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help! consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

NERSC job script generator tool



Today’s Agenda

- How jobs work
- What *must* I request?
- My first job
- **What else *can* I request?**
- Where is my job?
- Working interactively
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

What else *can* I request?

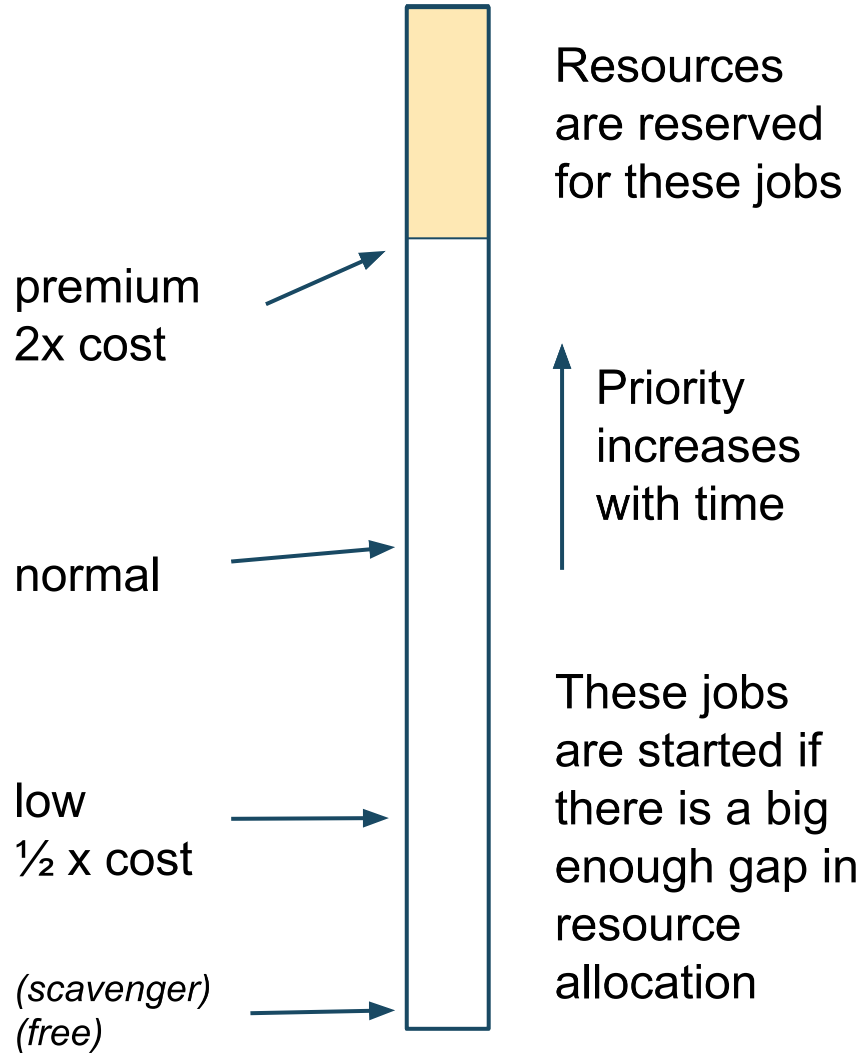
- **Quick vs cheap (QOS)**
  - Spend more to jump the queue?
    Or wait longer to spend less?
- **Faster I/O (Burst Buffer - Cori only)**
  - Cori has 1.8PB of SSD-based "Burst Buffer" to support I/O intensive workloads
  - Jobs can request a job-temporary BB filesystem, or a persistent (up to a few weeks) reservation
- **Containerized runtime environment (Shifter)**
  - Docker images can be pulled to the NERSC Shifter Image Gateway and used as an alternate, portable runtime environment for a job
  - Performance benefits for some dynamically-linked executables (especially Python)

QOS (Quality of Service)

- **For jobs in “-p regular”**
- **Affects where in queue your job starts**
  - And the cost! (NERSC-hours)

```
#SBATCH --qos=premium

#SBATCH --qos=low
```



What else *can* I request?

- Quick vs cheap (QOS)
  - Spend more to jump the queue?
    Or wait longer to spend less?
- **Faster I/O (Burst Buffer - Cori only)**
  - Cori has 1.8PB of SSD-based “Burst Buffer” to support I/O intensive workloads
  - Jobs can request a job-temporary BB filesystem, or a persistent (up to a few weeks) reservation
- Containerized runtime environment (Shifter)
  - Docker images can be pulled to the NERSC Shifter Image Gateway and used as an alternate, portable runtime environment for a job
  - Performance benefits for some dynamically-linked executables (especially Python)

Burst Buffer

- **Cori has 1.8PB of SSD-based "Burst Buffer" to support I/O intensive workloads**
  - Multiple read/write
  - Time-critical read/write (eg writing checkpoint files)
- **Stage-in/stage-out: data is moved before/after the job**
  - Can move individual files, or all files in a directory
- **More on BB tomorrow**

Burst Buffer

- **Job-temporary filesystem:**
  - Note “#DW” not “#SBATCH”
  - Also note environment variable in #DW directive - special case! (SLURM normally cannot expand environment variables in directives)

```
#DW jobdw capacity=100GB access_mode=striped type=scratch pool=sm_pool
#DW stage_in source=/global/cscratch1/sd/username/path/to/dirname
destination=$DW_JOB_STRIPED type=directory
#DW stage_out source=$DW_JOB_STRIPED/filename
destination=/global/cscratch1/sd/username/path/to/filename type=file
```

- **Access BB directory in job script:**

```
cd $DW_JOB_STRIPED
```

Burst Buffer

- **Persistent reservation:**
  - Good for multi-job workflow
  - Not reliable storage!
  - Note creation and deletion use “#BB” not “#DW” or “#SBATCH”
- **Create:**

```
#BB create_persistent name=myBBname capacity=100GB access=striped
type=scratch
```

- **Delete:**

```
#BB destroy_persistent name=myBBname
```

- **Use existing:**

```
#DW persistentdw name=myBBname
```

- **Access BB directory in job script:**

```
cd $DW_JOB_PERSISTENT
```

What else *can* I request?

- Quick vs cheap (QOS)
  - Spend more to jump the queue?
    Or wait longer to spend less?
- Faster I/O (Burst Buffer - Cori only)
  - Cori has 1.8PB of SSD-based "Burst Buffer" to support I/O intensive workloads
  - Jobs can request a job-temporary BB filesystem, or a persistent (up to a few weeks) reservation
- **Containerized runtime environment (Shifter)**
  - Docker images can be pulled to the NERSC Shifter Image Gateway and used as an alternate, portable runtime environment for a job
  - Performance benefits for some dynamically-linked executables (especially Python)

Shifter

- Docker-like container environment for HPC
- Works with MPI, with $SCRATCH, with Burst Buffer, etc
- Solves performance issues relating to dynamically loaded libraries

- More on Shifter tomorrow

Shifter

1. **Create Docker image**
2. **Push it to DockerHub**
3. **Pull it to NERSC Shifter ImageGateway**

   (Not currently available on Cori, use Edison for this step)

```
module load shifter
shifterimg -v pull docker:image_name:image_version_tag
shifterimg images       # list available images
```

4. **Use it in a job**

```
#/bin/bash -l
#SBATCH -N 2
#SBATCH -t 30
#SBATCH -L project,cscratch1
#SBATCH --image=docker:image_name:image_version_tag
#SBATCH --volume="/global/project/projectdirs/mpccc:/input;/global/cscratch1/sd/username/path/to/output:/output"

cd /output
srun -n64 shifter python ./my_python_app.py < /input/input_file.txt
```

Today's Agenda

- How jobs work
- What *must* I request?
- My first job
- What else *can* I request?
- **Where is my job?**
- Working interactively
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

Where is my job?

```
elvis@nersc:~> sqs
JOBID   ST REASON USER   NAME        NODES USED REQUESTED ...
2774102 R  Prolog elvis myscript.q 2     0:00 30:00

                  ...        SUBMIT              PARTITION RANK_P RANK_BF
                             2016-11-18T11:24:20 debug     N/A    N/A


elvis@nersc:~> ls -lt
total 11280
-rw-r----- 1 elvis elvis 132 Nov 18 11:24 slurm-2774102.out
-rw-r----- 1 elvis elvis 208 Nov 18 11:24 myscript.q
```

Where is my job?

https://my.nersc.gov/queues.php?machine=cori&full_name=Cori

My NERSC

- sleak
- Dashboard
- Queues
  - Cori
  - Edison
  - PDSF
  - Backlogs
  - Cori KNL Access Application
- Center Status
- File Browser
- Jobscript Generator
- Completed Jobs
- My Tickets
- Data Dashboard
- NX Desktop
- Changelog
- NERSC Homepage

Cori

Current Queue

10 records per page   Search: username repo etc.

Job ID	User	Nodes	Status	Queue	Submit Time	Time Required	Time Used	Priority Rank	Backfill Rank	QOS	Reason
3828807	azamat	2048	Queued	knl	2017-02-21T19:35:37	3:00:00	0:00	0001	0001	normal	Resources
3832873	plevine	30	Queued	regular	2017-02-22T13:34:12	36:00:00	0:00	0002	0002	premiu	Resources
3833023	zli	18	Queued	debug	2017-02-22T14:04:15	30:00	0:00	0003	0003	debug	Resources
3833037	ssn492	56	Queued	debug	2017-02-22T14:08:10	30:00	0:00	0004	0004	debug	Resources
3833040	lgerhard	20	Queued	debug	2017-02-22T14:09:05	30:00	0:00	0005	0005	debug	Resources
3833054	jialin	24	Queued	debug	2017-02-22T14:12:34	10:00	0:00	0006	0006	debug	Resources
3833071	borislo	16	Queued	debug	2017-02-22T14:15:46	20:00	0:00	0007	0007	debug	Resources
3833073	miller86	1	Queued	debug	2017-02-22T14:15:55	30:00	0:00	0008	0008	debug	Resources
3832045	tkurth	1024	Queued	knl_reboot	2017-02-22T11:41:02	8:00:00	0:00	0009	0009	normal	Resources
3828177	tkurth	256	Queued	knl_reboot	2017-02-21T14:10:56	1:00:00	0:00	0010	0010	normal	Resources

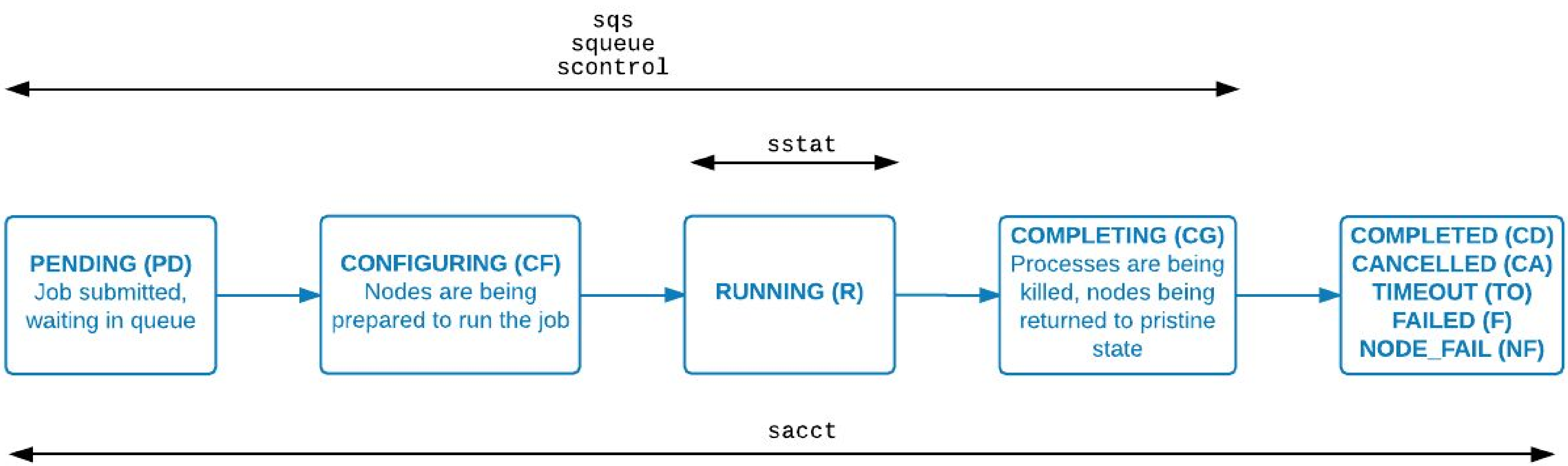
Showing 1 to 10 of 3,477 entries   Previous 1 2 3 4 5 … 348 Next

Toggle Columns: Job Name | Repo | Nodes | Queue | Submit Time | Time Required | Time Used | Priority Rank | Backfill Rank

Click here for the Classic Queue Look

Where is my job?

Job states you might see:



Note: completed jobs are not visible from sqs

Use sacct

Key Points

- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help! consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

Today's Agenda

- How jobs work
- What *must* I request?
- My first job
- What else *can* I request?
- Where is my job?
- **Working interactively**
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

Working interactively

```
elvis@nersc:~> salloc -N 2 -p debug -t 30 -C $CRAY_CPU_TARGET
salloc: Pending job allocation 334467
salloc: job 334467 queued and waiting for resources
salloc: job 334467 has been allocated resources
salloc: Granted job allocation 334467
salloc: Waiting for resource configuration
salloc: Nodes nid00[281-282] are ready for job
elvis@nid00281:~>
```

```
elvis@nid00281:~> srun -n 4 uname -n
nid00281
nid00282
nid00281
nid00282
elvis@nid00281:~>
```

- **"salloc" to start an interactive session on compute nodes**
- **Takes the same arguments as sbatch**
- **Still a batch job!**
  - Use "-p debug" to minimize waiting time
- **Inherits environment (eg modules you have loaded)**
- **Not a login shell!**
  - Run "source ~/.bash_profile" to get aliases etc
- **"srun" to run a command across all my nodes**

Today's Agenda

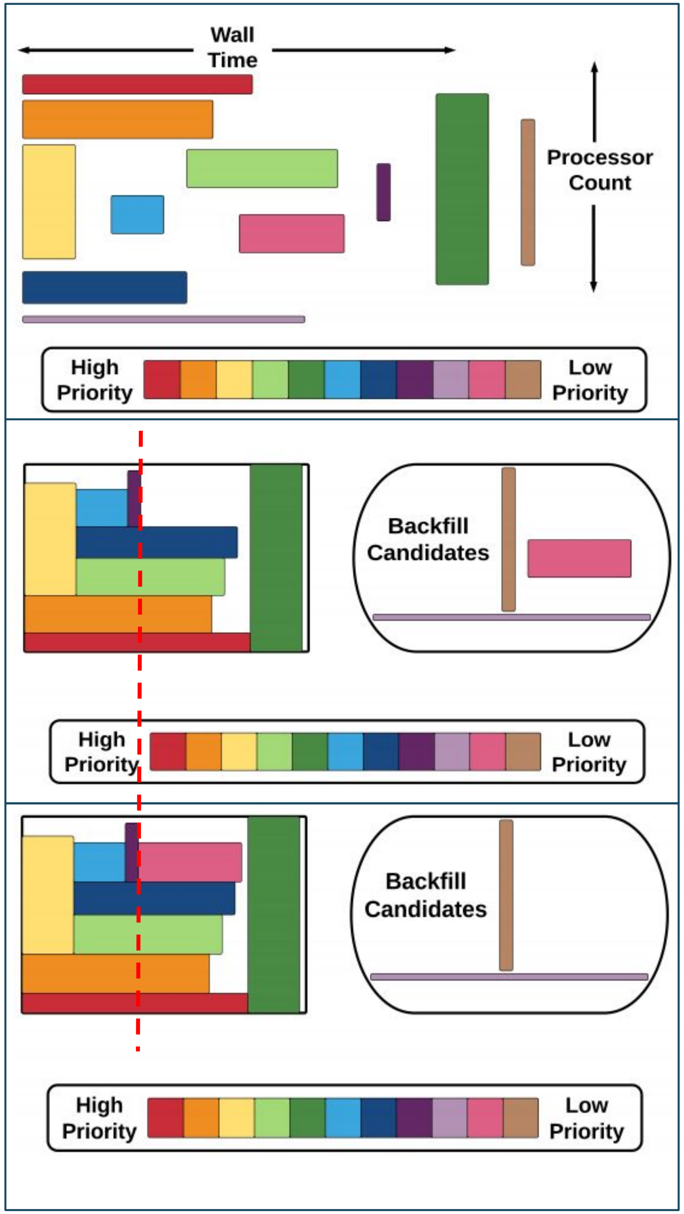
- How jobs work
- What *must* I request?
- My first job
- What else *can* I request?
- Where is my job?
- Working interactively
- **Getting through the queue faster**
- How usage is charged
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

How scheduling works

- **Jobs scheduled in two passes:**
  - Priority pass: highest-priority jobs are scheduled ASAP
  - Backfill pass: remaining jobs are scanned to find jobs that can start now in a gap left by priority pass

- **Short jobs are better backfill candidates than long jobs**
  - *Even if they require many nodes*

- **Jobs that can't run in backfill take several days to accumulate sufficient priority for the priority pass**



Queue wait time example

Edison - 2017 so far

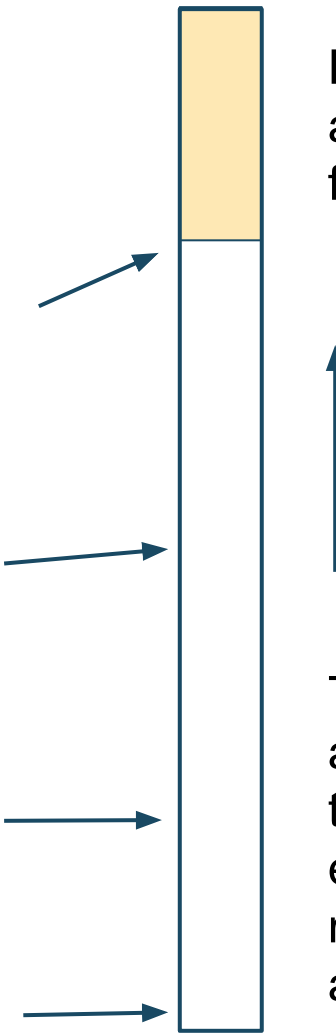
	Hours Requested																																																	
Nodes	<1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	48+
1	2.5	5.5	5.0	6.2	3.0	3.9	7.1	12	12	20	2.8	10	12	19	115	2.7	13		2.2		4.2			7.7	33	0.4					25	244	4.7	49	60	40	34				0.0				0.0		7.6	0.1	3.4	182
2	0.4	9.3	2.8	12	8.8	3.0	20	13	25	3.7	7.0	1.7	18	3.6	380	59	40				55			54	71		0.1			55	412		355		0.0	18	456													117
3	1.6	4.4	1.5	1.8	13	4.0	11	0.4	55	0.4	3.6		24			12	44		9.5		19		67		221				47		101		79				195													303
4	0.5	4.2	15	7.0	11	24	15		3.4	11	12	11	18	258	9.0	1.4			101	11	94		54		35	89				41	26						114													195
5	0.4	1.8	3.9	18	6.5	5.5	15	1.0	7.0	48	41	0.1	42				119		47		0.7	1.2	114	3.6	48								103				128													
6	3.8	12	1.4	3.7	13	3.4	4.2		17		11		17	104	462		84			90					185						22		72				184													232
7	0.1	7.6	3.9	7.9	38	5.1	60		50	30	32	19	22		105	69	56	76	56	167	74	53			42												133													
8	0.5	2.8	2.6	5.7	4.4	6.6	14	19	5.3		15	40	13			59			26		33			40	76	48					66						157													402
9	0.1	7.6	4.9	20	0.0	63			56		15		6.0												20					59							20													
10	2.0	3.1	14	21	9.8	55	30	40	19	21	28	94	34		70	113	31		67						17	18											76													406
11	3.6	0.6	1.9	1.0	5.1	0.0	6.9	2.8	1.6		34	144	30								10				10	8.0											238													
12	0.1	1.0	1.4	1.8	3.3	24	30	79	86	49	52	24	12				243								58												143													
13	0.1			0.0	2.1		41		30	11																																								
14	2.9		0.6																																															
15	3.8	0.5	7.5	15	8.5	73	28	81	71	92			42			40	128				2.7					48	34				35					98	91													
16	1.3	9.8	19	23	8.8	38	21	50	92		2.9		31		110		69		148		55				16												49													
17-19	0.3	0.4	0.4	0.3	42	38	43	170		123			13												74																									
20-23	0.4	1.1	3.8	5.2	13	31	35	100	25	16	47	14	33	100	18		67		13		53				56											104	61													
24-31	1.7	7.4	12	39	11	43	41		35	17	56	74	47	135			91		50		107				50		33		0.1		84						75													
32-47	2.1	9.1	21	12	27	11	37		42		67		64		66	94	37		91		62		34		64		79		9.9		2.4		106				59													
48-63	1.3	3.7	1.9	6.9	6.7	58	7.8	334	36		25		52								112				73				17							107	93													
64-127	2.2	18	15	30	3.8	26	11	5.4	30	61	65	120	92		75		77		60		107			65	80											264	201													
128-255	4.4	10	20	11	20	43	53		65	11	67		46	109		125	62				160			9.0	102						227		143				137													
256-511	4.9	34	110	17	22	21	47	117	0.9	139	29					57	166		64		180				188												413													
512-1023	7.6	39	36	39	56		66	0.8	2.0	60	55		77	176	33					137	448		1.8		112	134											53													
1024-1535	20	33	81	58	59	56	64	74			205		54								162				174																									
1536-2047	12	10	45				67			1.3															0.8												1,086													
2048-3071	35	30	67		18		58																																											
3072-4095	12		92																																															
4096-6143	33	336	416	104																																														
6144-9531																																																		

QOS Reminder

- **For jobs in "-p regular"**
- **Affects where in queue your job starts**
  - And the cost! (NERSC-hours)

```
#SBATCH --qos=premium
```

```
#SBATCH --qos=low
```



Resources are reserved for these jobs

premium
2x cost

Priority increases with time

normal

These jobs are started if there is a big enough gap in resource allocation

low
½ x cost

(scavenger)
(free)

Key Points

- HPC work is via batch system
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- Key commands:
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- Watch your budget! NERSC-hours and charge factors
- **Help! consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

Today's Agenda

- How jobs work
- What *must* I request?
- My first job
- What else *can* I request?
- Where is my job?
- Working interactively
- Getting through the queue faster
- **How usage is charged**
- Which cores are running what? (advanced)
- Workflows - job arrays and dependencies

How usage is charged

"NERSC-hours" - based on performance/node relative to 1 core of Hopper (past NERSC system)

1 wallclock-hour is many NERSC-hours!

System	Node Architecture	Base Charge per Node Hour (NERSC Hours)	System Size (Nodes)	Cores per Node
Cori	Intel Xeon Phi (KNL)	96*	9,303	68
Cori	Intel Xeon (Haswell)	80	2,004	32
Edison	Intel Xeon (Ivy Bridge)	48	5,576	24

	System	Charge Modification
Premium Queue Priority	Cori and Edison	2 x base charge
Low Queue Priority	Cori and Edison	0.5 x base charge
Big Job Discount*	Edison jobs that use >683 nodes	0.6 x base charge
Shared Node	Cori Haswell nodes	2.5 NERSC hours per core hour
Scavenger	Cori and Edison	No charge ("free")
Cori Xeon Phi Pre-Production	Cori Xeon Phi (KNL) nodes	No charge ("free") until production computing begins on July 1, 2017

How usage is charged

Your repo is charged for each node your job was allocated, for the entire duration of the job:

```
#SBATCH -N 64
#SBATCH -t 1:00:00
srun -N 32 -t 30 ./my_long_app.x
```

- App only ran on 32 nodes, but 64 nodes were requested (and allocated), so you are charged for 64 nodes
- 1 hour was requested, but the job ended after 30 minutes, so you are charged for 30 minutes

If you have access to more than one repo, specify which to charge in your batch script:

```
#SBATCH -A repo_name
```

Key Points

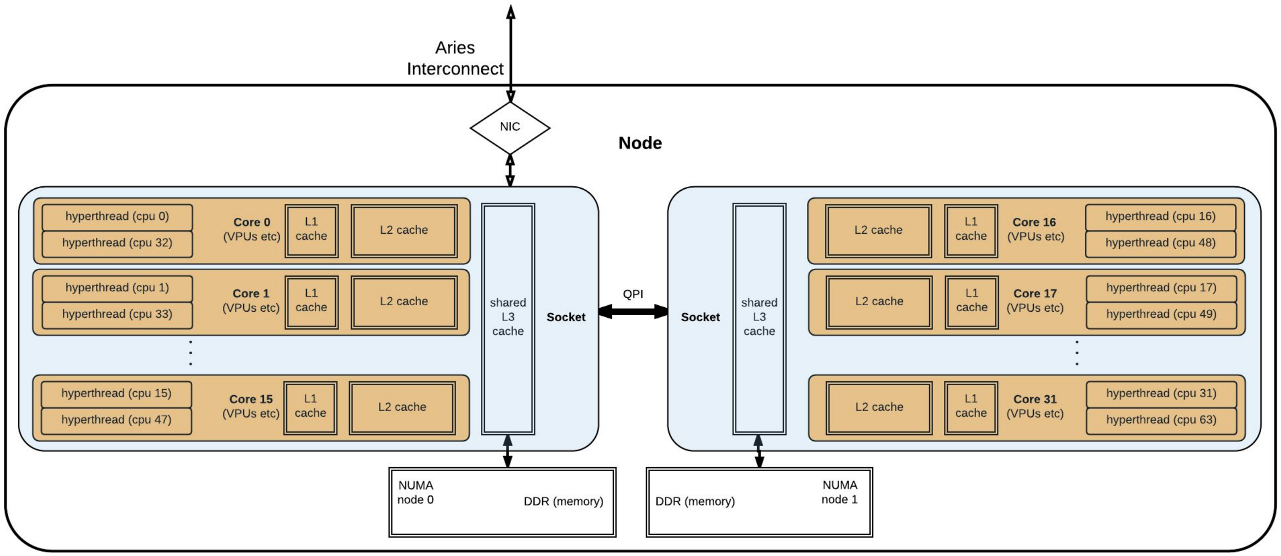
- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help! consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

Today's Agenda

- How jobs work
- What *must* I request?
- My first job
- What else *can* I request?
- Where is my job?
- Working interactively
- Getting through the queue faster
- How usage is charged
- **Which cores are running what? (advanced)**
- Workflows - job arrays and dependencies

Process and memory affinity



Modern compute nodes have multiple sockets, cores, hyperthreads and Non-uniform memory access (NUMA)

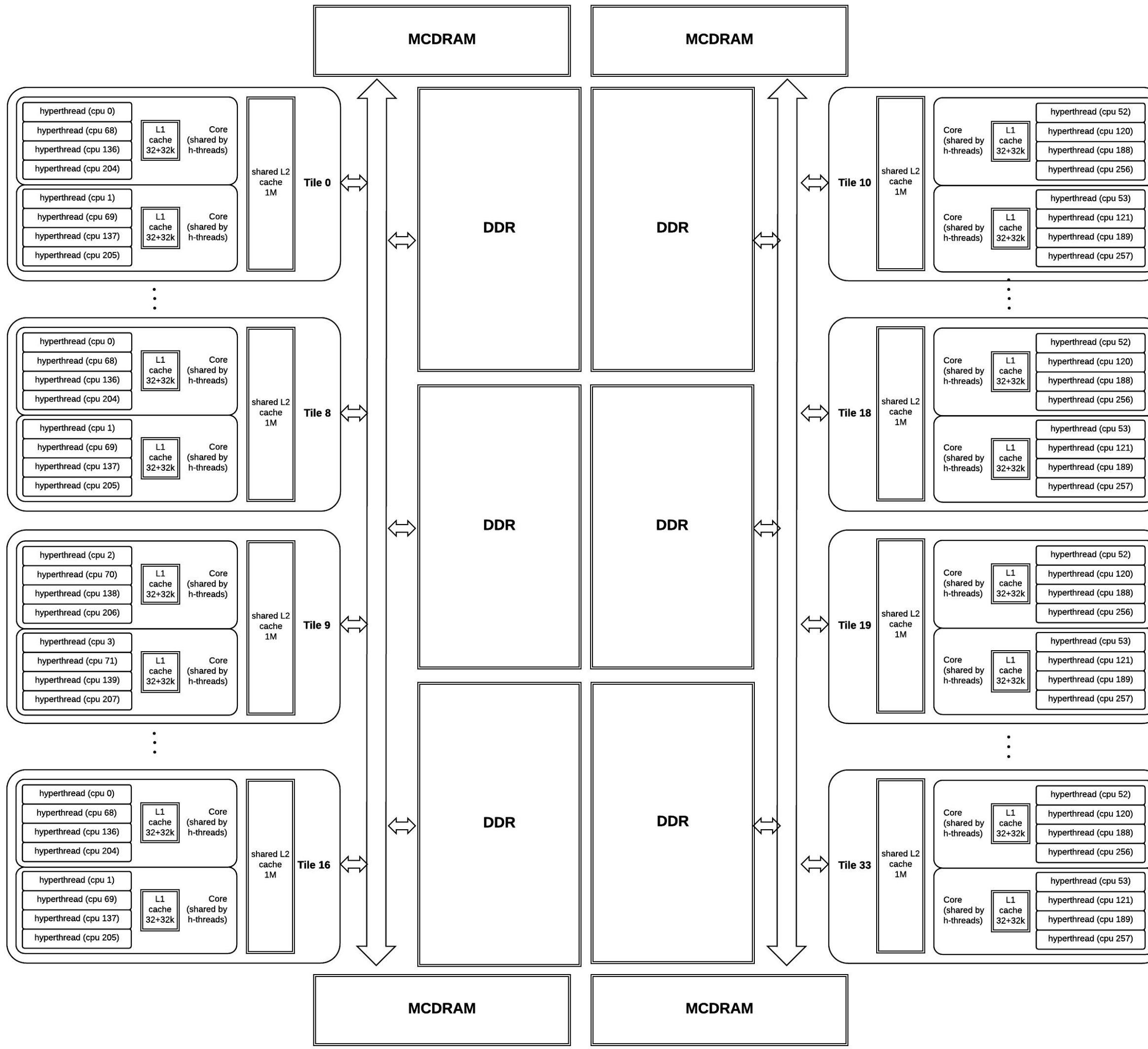
- Two tasks (OpenMP thread or MPI ranks) using hyperthreads of same core are contending for arithmetic, cache resources
- A task using memory from opposite socket has reduced memory bandwidth
- Two OpenMP threads on opposite sockets might cause cache thrashing between the two L3 caches

Process and memory affinity

KNL node is even more complex

- Tiles as well as cores and hyperthreads
- Up to 8 NUMA nodes for tasks and memory to land on

Thread placement and memory affinity are increasingly important for good performance



Process (task) affinity

```
srun -n 64 -c 4 ./my_exec
```

- **The "-c" sbatch/srun option controls number of CPUs reserved per task, *not* task placement or binding!**
- **Linux will place threads wherever it sees fit, eg "cram them into the fewest possible number of cores, leaving other cores empty" (not ideal!)**
- **If ("-n" * "-c" != total_available_cpus) then SLURM+Linux can get confused => pathologically bad placement**

- **Solution: use --cpu_bind:**

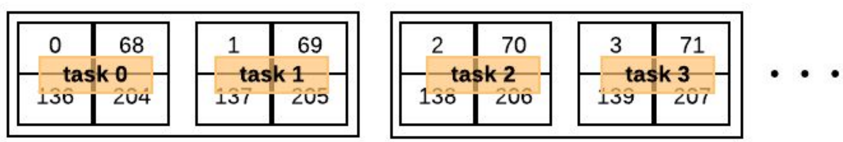
```
srun -n  64 -c 4 --cpu_bind=verbose,cores   ./my_exec
srun -n 128 -c 2 --cpu_bind=verbose,threads ./my_exec
```

Process (task) affinity

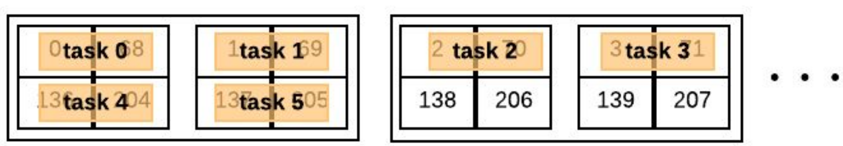
Solution: use --cpu_bind:

```
srun -n  64 -c 4 --cpu_bind=verbose,cores   ./my_exec
srun -n 128 -c 2 --cpu_bind=verbose,threads ./my_exec
```

- **Controls what a task (MPI rank) is bound to**
  - If no more than 1 MPI rank per core: `--cpu_bind=cores`

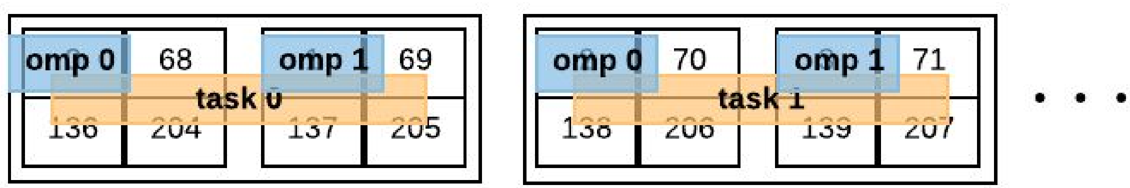


  - If more than 1 MPI rank per core: `--cpu_bind=threads`



Thread affinity (OpenMP)

```
export OMP_NUM_THREADS=2
export OMP_PROC_BIND=spread     # or close
export OMP_PLACES=cores         # or threads, or sockets
srun -n  32 -c 8 --cpu_bind=verbose,cores   ./my_exec
```



...If using hyperthreads, use OMP_PLACES=threads

Memory affinity

Linux default behavior is to allocate to closest NUMA-node, if possible

Not always optimal:

- **KNL nodes: DDR is "closer" than MCDRAM**

```
#SBATCH -C knl,quad,flat
export OMP_NUM_THREADS=4
srun -n16 -c16 --cpu_bind=cores --mem_bind=map_mem:1 ./a.out
```

- **NUMA node 1 is MCDRAM in quad,flat mode**
- **"Mandatory" mapping: if using >16GB, malloc will fail**

Memory affinity

"Preferred" affinity: if preferred NUMA node is full, allocate to another NUMA node

- Not yet supported by SLURM
- Use numactl instead

```
module load numactl
srun -n16 -c16 --cpu_bind=cores numactl -p 1 ./a.out
```

Today's Agenda

- How jobs work
- What *must* I request?
- My first job
- What else *can* I request?
- Where is my job?
- Working interactively
- Getting through the queue faster
- How usage is charged
- Which cores are running what? (advanced)
- **Workflows - job arrays and dependencies**

Job arrays

Need to run a set of almost-identical jobs?

```
#!/bin/bash -l
#SBATCH -N 64
#SBATCH -t 6:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quat,cache

runid=19 # update before each job
mkdir $SCRATCH/runs-$runid
cd $SCRATCH/runs-$runid
export OMP_NUM_THREADS=16
export OMP_PLACES=cores
export OMP_PROC_BIND=spread
srun -n 1024 -c 64
  --cpu_bind=cores ./a.out $runid
```

```
#!/bin/bash -l
#SBATCH -N 64
#SBATCH -t 6:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quat,cache
#SBATCH --array=1-100
runid=$SLURM_ARRAY_JOB_ID
mkdir $SCRATCH/runs-$runid
cd $SCRATCH/runs-$runid
export OMP_NUM_THREADS=16
export OMP_PLACES=cores
export OMP_PROC_BIND=spread
srun -n 1024 -c 64
  --cpu_bind=cores ./a.out $runid
```

Job arrays

- **Convenient way to manage sets of near-identical jobs**
- **The SLURM directives describe resources for a *single job in the array***
- **Appears in queue as, eg 1234567_1, 1234567_2,..**
- **To cancel an individual member:**

```
scancel 1234567_7   # cancel array member number 7
```

- **To cancel the whole array:**

```
scancel 1234567
```

Job array gotchas

- **The SLURM directives describe resources for a *single job in the array***
  - Common help ticket: “I wanted to run 1000 copies of my 1-node job for 1 hour, so I submitted this:”

```
#SBATCH -N 1000
#SBATCH -t 1000:00
#SBATCH --array=1-1000
```

  - Desired effect: uses 1000 node-hours (80000 NERSC-hours)
  - Actual effect: uses 1000x1000x1000 node-hours (80 billion NERSC-hours = whole year allocation for repo
    - … And P.I. is ***furious***

- **Lesson: test your script on a small job array first!**

Dependencies

Some workflows require a job to run only after another job has completed (or perhaps, only if another job fails)

```
elvis@nersc:~> sbatch job1.q
Submitted job 5436
elvis@nersc:~> sbatch -d afterok:5436 job2.q
```

Key Points

- **HPC work is via batch system**
  - Dedicated subset of compute resources
  - Login nodes are shared resource for building code, editing scripts, etc. Use batch jobs for real work
- **Key commands:**
  - sbatch / salloc - submit a job
  - srun - start an (optionally MPI) application within a job
  - sqs - check the queue for my job status
- **Queues are long!**
  - Work *with* the system to get better turnaround time
- **Watch your budget! NERSC-hours and charge factors**
- **Help!** **consult@nersc.gov**

www.nersc.gov/users/computational-systems/{cori,edison}/**running-jobs/**

NERSC

U.S. DEPARTMENT OF ENERGY

Office of Science

BERKELEY LAB

Lawrence Berkeley National Laboratory